»Лекционен курс

»ООП1 (Java)

Пакети ›

Регистрация

https://tinyurl.com/2bepln7u

Мотивация и предназначение

Мотивация

- В примерите досега името на всеки клас принадлежи към едно и също пространство на имена:
  - Това означава, че трябва да се използват уникални имена за всеки клас;
  - За избягване на колизии.
- При нарастване броя на класовете в една система, без някакъв начин за управление на пространството за имена, бихме били затруднени да следим уникалността на имената на класовете.
- Необходим е механизъм за групиране на логическите свързани класове заедно.

Пакети

» Java предоставя механизъм за разделяне на пространството на имена на класовете в по-лесно управляеми единици:

> Наречени пакети.

Предназначение

- Един пакет изпълнява две задачи:
  - Първо, той предлага механизъм, чрез който свързани части от програмата могат да бъдат организирани като една обща единица.
    - Класовете, определени в даден пакет, трябва да бъдат достъпни чрез името на пакета им.
    - По този начин един пакет предоставя начин за създаване на колекции от класове.
  - Второ, пакетът участва в механизма за контрол на достъпа в Java.
    - Класовете, определени в даден пакет, могат да бъдат направени частни (private) за този пакет и да не са достъпни с код извън пакета.
    - По този начин пакетът предоставя средство, чрез което класовете могат да бъдат капсулирани.

Пространство на имена

- Когато даваме име на клас, ние разпределяме име от пространството на имената.
- В Java класовете не могат да имат едно и също име в едно пространство на имена.
- В рамките на дадено пространство от имена, всяко име на клас трябва да бъде уникално.
- Примерите до сега използваха пространство от имена по подразбиране (глобално).

Пространство на имена

- Въпреки, че за кратки примерни програми това е приемливо, превръща се в проблем за реален софтуер:
  - Пространството от имена по подразбиране става препълнено и трудно проследимо;
  - Намирането на уникални имена за всеки клас става вече трудно.
- Освен това трябва да се избягват колизии с имена на класове, създадени от други програмисти, работещи по същия проект, и с библиотеките на Java.

Пространство на имена

- Решението на тези проблеми са пакетите: механизъм за разделяне пространството с имена.
- Когато един клас е в определен пакет, името на този пакет е прикрепено към всеки клас, като по този начин се избягват колизии с имена с други класове, които имат същото име, но са в други пакети.

Свързани с операционната система

- Пакетът е именована колекция от свързани класове, която може да служи като библиотека от класове за използване във всяка програма.
- С пакетите не е необходимо да поставяте всички тези класове в същата директория като нашата програма.
- Пакетите в Java са тясно свързани с организацията на файловата система:
  - Директории (папки);
  - Имена на пътища за папки;
  - Как операционната система използва променлива на път.
- Това не са теми на Java.

Работа с пакети

Достъп

- Тъй като един пакет обикновено съдържа свързани класове, Java дефинира специални права за достъп в рамките на пакета.
- В един пакет можем да определим код, който е достъпен от друг код в рамките на същия пакет, но не и от код извън пакета.
- Това ни позволява да създаваме самостоятелни групи от свързани класове с частен достъп.

Работа с пакети

```
package Package_Name;
```

- Пакетът е просто колекция от класове, които са групирани в папка.
- Името на папката е името на пакета.
- Класовете в пакета се поставят в отделни файлове.
- Всеки файл в пакета има следния ред в началото на файла:

Работа с пакети

```
package general.utilities;
```

- Празни редове или коментари могат да се появят преди този ред, но нищо друго не може да се появи преди него.
- Името на пакета обикновено се състои от малки букви.
- Обикновено две имена се разделят с точка.
- Напр., ако general.utilities е името на пакета, всеки от файловете в пакета ще има следния израз в началото на файла:

Работа с пакети

```
import general.utilities.HelpfulClass;
```

- Всяка дефиниция на програма или клас може да използва всички класове в пакета, като постави подходящ оператор за импортиране в началото на файла, съдържащ дефиницията на програмата или класа.
- Това е вярно, дори ако дефиницията на програма или клас не е в същата директория като класовете в пакета.
- Напр., ако искаме да използваме класа HelpfulClass, който е в пакета general.utilities, ще поставим следното в началото на файла (задаваме името на класа, а не името на файла, който е HelpfulClass.java):

Пакет по подразбиране

- Всички класове в Java принадлежат към някакъв пакет.
- Когато не е специфициран пакет, се използва пакетът по подразбиране (глобален).
- Освен това пакетът по подразбиране няма име, което прави пакета по подразбиране прозрачен.
- Ето защо досега не е трябвало да се притесняваме за пакетите.
- Въпреки, че пакетът по подразбиране е добър за кратки, примерни програми, той не е подходящ за реални приложения.
- В повечето случаи ще е необходимо да дефинираме един или повече пакети за нашия код.

Дефиниране на пакети

Дефиниране на пакети

- Създаването на пакет е съвсем лесно: просто включваме ключовата дума package като първи оператор в кода.
- Всички класове, декларирани в този файл, ще принадлежат към посочения пакет.
- Една декларация на пакет дефинира пространство на имена, в което се съхраняват класовете.
- Ако не е дадено име на пакет, имената на класовете се поставят в стандартния пакет:
  - Пакет без име;
  - Добър за малки примерни програми, но неадекватен за реални приложения.

Именуване на пакети

- Подобно на останалата част от Java, имената на пакетите са чувствителни към малки и големи букви.
- Обикновено за имена на пакети се използват малки букви.

Java-синтаксис

Синтаксис на клас в пакет

```
<Blank lines or comments.>
package Package_Name;
<A class definition.>
```

Примери

```
package general.utilities;
package java.io;
```

Синтаксис на import оператора

```
import Package_Name.Class_Name_Or_Asterisk;
```

Задаване име на клас импортира само един клас от пакета.
* импортира всички класове в пакета.

Примери

```
import java.util.Scanner;
import java.io.*;
```

Именуване на пакети

- Името на пакета не е произволен идентификатор.
- Той казва на компилатора къде да намери класовете в пакета.
- Всъщност името на пакета казва на компилатора името на пътя за директорията, съдържаща класовете в пакета.
- За да намери директорията за пакет, Java се нуждае от две неща:
  - Името на пакета;
  - Директориите, изброени в стойността на променливата на пътя на класа.

Пакети и файлова система

Пакети и файлова система

- Стойността на променлива на пътя на класа казва на Java къде да започне търсенето на пакет.
- Променливата на пътя на класа не е променлива на Java.
- Това е променлива, която е част от операционната система и съдържа имената на пътищата на списък с директории.
- Когато Java търси пакет, започва търсенето в тези директории.
- Нека наречем тези директории базови.

Пакети и файлова система

- Името на пакета указва относителното име на пътя за директорията, която съдържа класовете на пакета.
- Това е относително име на път, тъй като предполага, че стартирате в основната директория на пътя на класа и следвате пътя на поддиректориите, дадени от името на пакета.

Пакети и файлова система

```
\myjavastuff\libraries
```

Да предположим напр., че базовата директория на пътя на класа (операционна система може да използва “/” или “\”):

```
\myjavastuff\libraries\general\utilities
```

Да предположим, че класовете са в директорията:

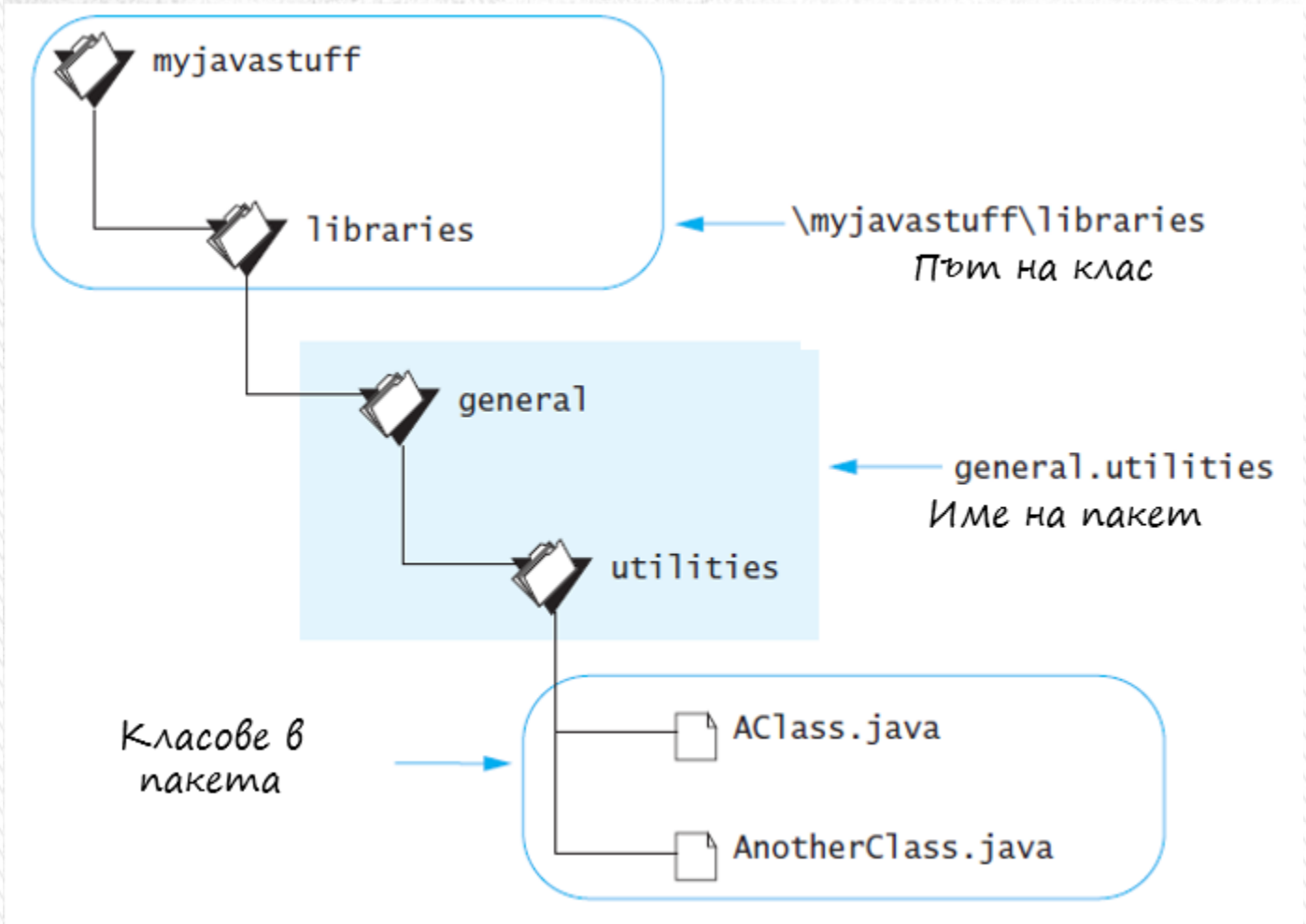
```
general.utilities
```

В този случай пакетът трябва да бъде именуван.

Пакети и файлова система

- Името на пакета представлява списък с директории, водещи от базовата директория към класовете на пакета.
- По този начин името на пакета казва на Java през какви поддиректории да премине, започвайки от директорията на базовия клас, за да намери класовете на пакета.
- Тази организация е изобразена на следващия слайд
  - Точката в името на пакета означава по същество същото като “\” или “/” (в зависимост от използваната операционна система).

Общ вид


myjavastuff
libraries
\myjavastuff\libraries
Път на клас
general
general.utilities
Име на пакет
utilities
Класове в пакета
AClass.java
AnotherClass.java

Базова директория

- Указваме основните директории на пътя на класа, като задавате променливата за пътя на класа (средата) променлива.
- Начинът, по който задаваме променливата на пътя на класа, зависи от операционната система.
- Напр., в Windows използваме контролния панел за да зададем или създадем променлива на средата с име CLASSPATH.

Видимост

Компилационни единици

- Компилационна единица:
  - Файл с изходен код на Java;
  - Притежава име (еднакво като това на класа), завършващо с .java;
  - Само един public клас.
- При компилиране на .java файл
  - Получаваме изходен файл с точно същото име, но с разширение .class;
  - Всеки .java файл кореспондира със съответен .class файл.

Импорт

- Ако искаме да използваме един клас е даден пакет можем да използваме пълно квалифицирано име
  - Пример: x = java.lang.Math.sqrt(5);
- В много случаи такова използване е неудобно.
- Java представя възможност за специфициране на достъп до определени класове от даден пакет посредством import маханизъм:
  - Импортирането предоставя механизъм за управление на “пространства от имена”.

Пример за импорт

```
java.util.Date d = new java.util. Date();
java.awt.Point p = new java.awt.Point(1,2);
java.awt.Button b = new java.awt.Button();
```

```
import java.util.Date;
Import java.awt.*;

...
Date d = new Date();
Point p = new Point(1,2);
Button b = new Button();
```

Пакетът java.lang

- Пакетът е по подразбиране.
- Java предполага, че класовете, съдържащи се в java.lang пакета са винаги налични:
  - Т.е. все едно, че import java.lang.* наличен в началото на всяка програма.

Йерархия от пакети

- Java използва директории на файловата система за съхраняване на пакети.
- Можем да създаваме йерархия на пакетите.
  - За целта просто разделяме името на пакет с тази над него с помощта на “.”.
- Обща форма: package alpha.beta.gamma;
  - Пример: package alpha.beta.gamma.
  - Трябва да бъде съхранен в директория „... / alpha / beta / gamma“, където „...“ указва пътя към специфицираните директории.

Има ли значение къде ще бъде поместен един клас?

» Името, което се използва за реферирane на класа

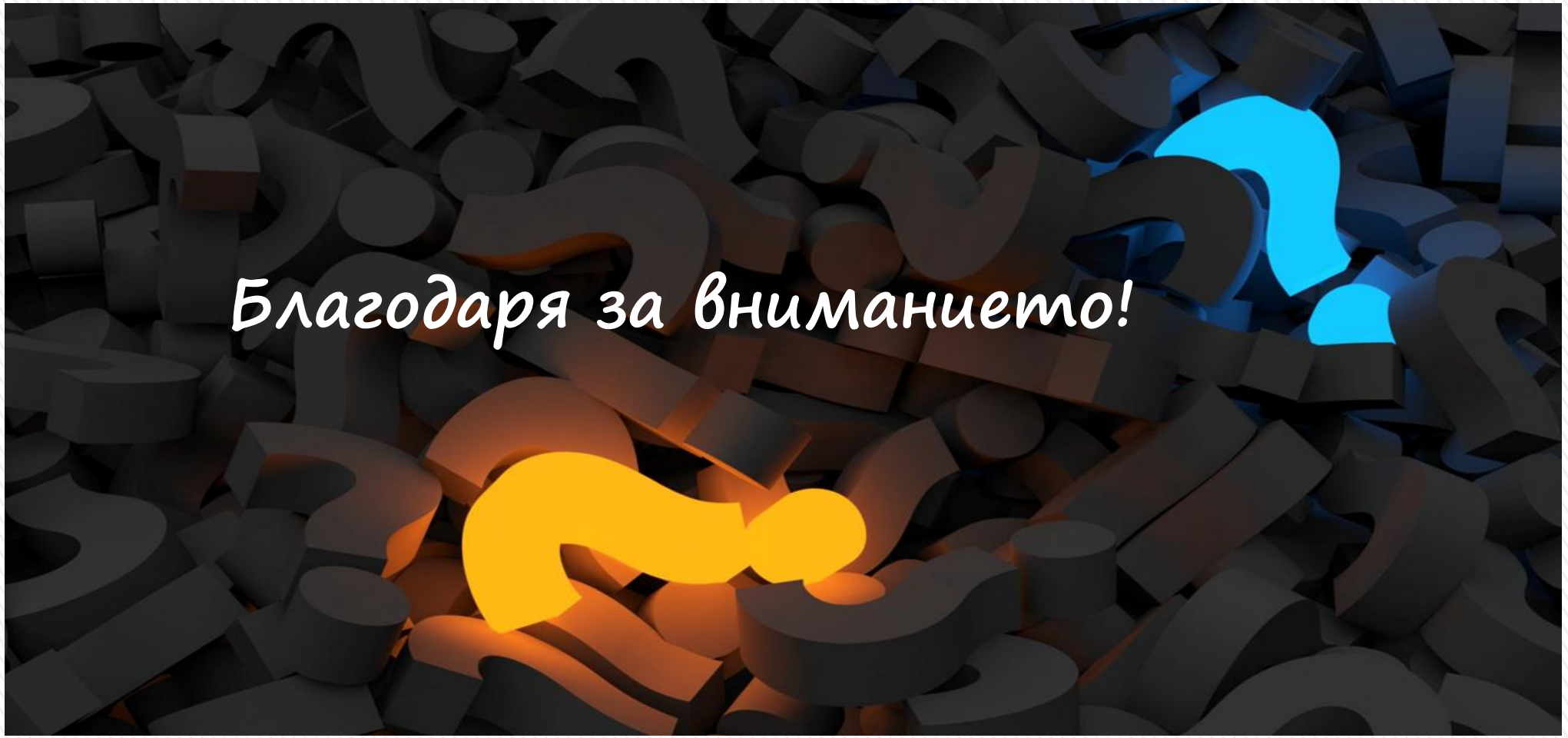
Видимост на променливите на инстанции

- Какво става с видимостта на променливите на инстанции
  - public
  - private
- Какво става ако не е декларирана нито като public нито като private?

По подразбиране е видима в пакета – т.е. в методите на класовете, които принадлежат към същия пакет.

Видимост на класове

- Един клас, деклариран като public е видим за всички класове.
- Един клас, без модификатор за достъп е видим за всички класове в същия пакет.
- Един клас, деклариран като private, protected:
  - Вътрешни класове.


Благодаря за вниманието!